Babraham
Bioinformatics

Exercises:
Analysing RNA-Seq data

Version 2019-04

Licence

Introduction

In this session we will go through all of the steps of a simple RNA-Seq differential expression analysis which will include:

- Mapping an example dataset to a reference genome with HiSat2
- Visualisation of mapped reads in SeqMonk
- Quality control of mapped data
- Quantitation of reads against genes
- Differential expression analysis in SeqMonk with DESeq
- Differential expression filtering using Intensity Difference in SeqMonk

Software

The software which will be used in this session is listed below. Software which requires a linux environment is indicated by an asterisk*:

- HiSat2* (https://ccb.jhu.edu/software/hisat2)
- Samtools (http://www.htslib.org/)
- FastQC (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/)
- Python (http://www.python.org/)
- Cutadapt (https://pypi.python.org/pypi/cutadapt)
- Trim_Galore (http://www.bioinformatics.babraham.ac.uk/projects/trim_galore/)
- SeqMonk (http://www.bioinformatics.babraham.ac.uk/projects/seqmonk/)
- R (http://www.r-project.org/)
- DESeq2 – part of bioconductor (http://www.bioconductor.org/)

Data

The data in this practical comes from:

- Yeast data for mapping comes from GEO accession SRR453566 (actual data is a subset of the full data at this accession)
- Yeast genome data and GTF models come from Ensembl (http://www.ensembl.org/Saccharomyces_cerevisiae/)
- Mouse RNA-Seq data for differential expression are selected RNA-Seq samples from GEO accession GSE48364

Part1: Raw sequence processing

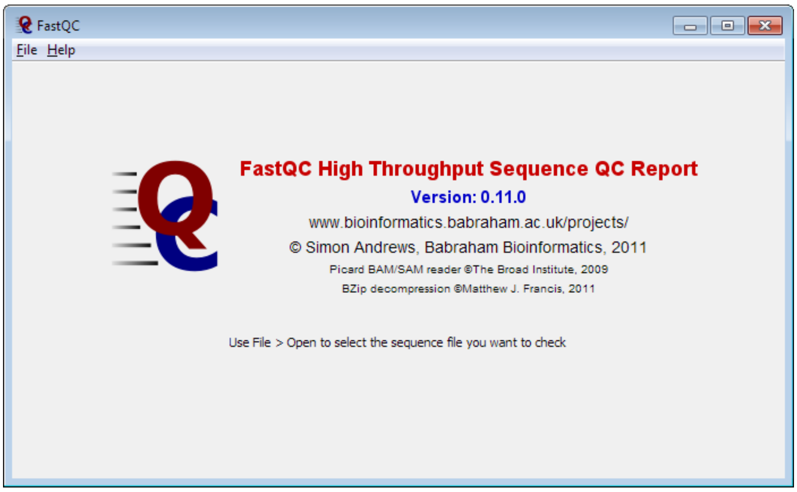
Exercise 1: Quality Control – Run QC on the FastQ file from the sequencer

In this section we will run a standard (non-RNA-Seq specific) QC pipeline on the data we are going to map so we can be sure that the data we're using doesn't have any obvious systematic problems before we continue with the analysis. The data for this part should be in the "Yeast_data_for_mapping" subfolder of the main data folder.

FastQC can be run from the command line or as a graphical application. Here we are going to run it as a graphical application so you can start the program by running:

```
fastqc &
```

The `&` on the end means that the program will run but it won't block you from running other commands in your terminal. If you didn't add this then you'd need to wait until `fastqc` had exited to be able to run any other commands. You will see a window open up:



FastQC High Throughput Sequence QC Report

Version: 0.11.0

www.bioinformatics.babraham.ac.uk/projects/

© Simon Andrews, Babraham Bioinformatics, 2011

Picard BAM/SAM reader ©The Broad Institute, 2009

BZip decompression ©Matthew J. Francis, 2011

Use File > Open to select the sequence file you want to check

You can select File > Open and then select the `SRR453566_yeast_rnaseq.fq.gz` file to launch a QC analysis of this file.

You can have a look through the QC results to try to answer the following questions:

- Did any of the QC modules trigger a warning or alert condition?
- Do the base call qualities provided by the sequencer suggest the data is high quality, or might it benefit from being quality trimmed?
- Are there any consistent sequence biases in the data? If so, can these be explained by the presence of contaminating sequences or other artefacts which are known to affect RNA-Seq datasets?
- Is there any suggestion of the presence of adapter sequence which might need to be removed?
- Does the duplication level of the data look reasonable given that this is RNA-Seq data?

Exercise 2: Trimming - Turn the raw fastQ file into a trimmed fastQ file containing only good quality data

In this section we will take the raw reads contained in the fastq file, trim them to remove adapter and low quality ends

Trim_Galore is run from the command line and is a wrapper around another program called Cutadapt. You can run Trim Galore with default options and it will try to detect the adapter that is present and remove it.

Trim Galore should produce a trimming report which you can have a look through to see details of any trimming that was carried out. An easier way of checking whether the trimming appears to have improved the data is to run FastQC again on the trimmed data. This can be incorporated into the original Trim Galore command by specifying `--fastqc` as an option, which will tell Trim Galore to run FastQC directly after trimming.

```
trim_galore --fastqc SRR453566_yeast_rnaseq.fq.gz
```

This should show some output on the screen and produce a file of trimmed data called `SRR453566_yeast_rnaseq_trimmed.fq.gz`

A FastQC html report should also be produced. You can view the html report in a web browser, e.g.:

```
firefox SRR453566_yeast_rnaseq_trimmed_fastqc.html
```

Exercise 3: Mapping – Take the data in the trimmed fastQ file and map it to a reference genome to create a BAM file of mapped positions

In this section we take the fastq output from exercise 2 and map it against the yeast genome. We will provide a set of gene models in a GTF file so that known splice junctions will be taken into account automatically and won't have to be re-discovered.

Creating a Genome Index

You will have been provided with a yeast genome sequence in a file called `yeast_genome.fa`. Before you can run hisat2 you will need to create an index file from this sequence which hisat2 can then use:

```
hisat2-build yeast_genome.fa yeast_index
```

The first option you supply here is just the name of the fasta file which contains your genome sequence. The second option is the base name for the index. This is the name you will need to supply when you run hisat2, and you should also now be able to see the following files in your directory:

```
yeast_index.1.ht2        yeast_index.4.ht2        yeast_index.7.ht2
yeast_index.2.ht2        yeast_index.5.ht2        yeast_index.8.ht2
yeast_index.3.ht2        yeast_index.6.ht2
```

You don't need to be concerned with the exact naming and number of files produced by the indexing. All that matters is the prefix name you supplied (`yeast_index`), as this is what you'll need to pass to the `hisat2` mapping program later.

Extracting splice sites from a GTF annotation file

Hisat2 can import a pre-processed set of known splice junctions to aid the mapping of your data. It requires its own special format for these files, rather than the standard GTF file format, but it also comes with a script to convert GTF files for you. To make your splice sites file you can run:

```
extract_splice_sites.py yeast_gene_models.gtf > yeast_splice_sites.txt
```

In this case the `extract_splice_sites.py` program takes only a single argument, the name of the gtf file containing the gene models. Normally the program would write its output to the screen. To save it to a file we use a `>` symbol to redirect the output to a file (`yeast_splice_sites.txt`). This program runs the only quick operation during mapping, and it should complete almost instantly.

You can check the top of the output file by running:

```
head yeast_splice_sites.txt
```

Running hisat2

Now you can run the mapping. The format of the hisat2 command we're going to use is shown below. We've split this over multiple lines here to make it clearer, but it should be entered as a single line command with spaces after the end of each line below:

```
hisat2 -x yeast_index
--known-splicesite-infile yeast_splice_sites.txt
-p 2
-U SRR453566_yeast_rnaseq_trimmed.fq.gz
| samtools view -bS - > SRR453566_yeast_rnaseq.hisat.bam
```

The options in this command are as follows:

`-x` : Specifies the index file to use – this is the basename for the index which you created in the "Creating a genome index" step

`--known-splicesite-infile` : Pre-populates the search with the locations of the splice sites you extracted from the GTF file when you ran `extract_splice_sites.py`.

`-p 2` : This tells hisat2 that it can use 2 CPU cores on the machine. Short read mapping scales very efficiently across cores so if you have more cores available then you should use them to speed up your analysis.

`-U` : Specifies the fastq file of sequence reads to map. If you had paired end data and therefore 2 input files you would use `-1` and `-2` to specify the two files instead of `-U`

`| samtools view -bS - >` : The final part of the command actually sends (pipes) the output of hisat into another program called samtools. The purpose of this is to compress the raw sam format output from hisat into the more compact bam format.

The mapping will take a few minutes to complete. Once it's finished you should see the mapping statistics printed in the console, and you should see the bam file of aligned reads.

If the command fails but doesn't exit so you can't enter more commands into your shell you can press `Control+c` to force it to quit.

Part 2: Visualisation, Exploration and Differential Expression

Exercise 4: Visualisation and Quantitation

In this exercise we're going to take a set of mapped data files from an older tophat pipeline and perform QC, quantitation and analysis of the mapped data. The data we're using are not the yeast data from before, but are a larger mouse dataset which would take too long to process in real time for this exercise.

The raw data for this part of the practical can be found in the `Mouse_Mapped_data` subfolder of the main data folder. The experiment consists of 6 datasets, but to save time we have pre-loaded 5 of these into a SeqMonk project already, and you just need to add the final one.

The SeqMonk software we're using for the rest of this practical can be run either under linux, windows or OSX.

Importing mapped data into SeqMonk

Open SeqMonk on your machine either by double clicking the SeqMonk icon or by typing '`seqmonk`' in a linux command shell.

Open the existing project into which 5 of the 6 datasets have already been imported.
File > Open Project

The project file is called `mouse_mapped_data.smk` and is in your `Mouse_Mapped_data` folder.

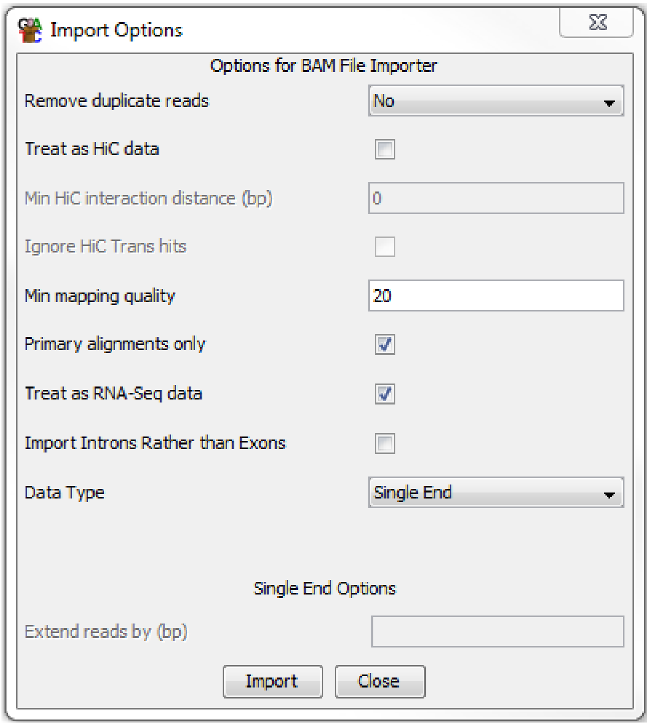
Once the project is open you can import your BAM file by selecting:

File > Import Data > BAM/SAM

Then selecting the `ESC1_GRCm38_hisat2.bam` file which is in the same folder.

The import options should have been detected automatically, but check that the sample has been flagged as RNA-Seq data, and that the library type is "Single End".

These default options will cause some filtering of the data to be carried out. The minimum mapping quality of 20 means that only high quality reads will be imported.



Import Options

Options for BAM File Importer

Remove duplicate reads: No

Treat as HiC data: □

Min HiC interaction distance (bp): 0

Ignore HiC Trans hits: □

Min mapping quality: 20

Primary alignments only: ☑

Treat as RNA-Seq data: ☑

Import Introns Rather than Exons: □

Data Type: Single End

Single End Options

Extend reads by (bp):

Examination of the raw data

In any sequencing analysis the first step in your analysis should be to spend a couple of minutes looking through the raw data to see if you can see any obvious problems. Look around the genome and see if you think this data is OK. Questions to focus on would be:

- Do the reads look like they are mapping to the annotated transcripts?
- Is there evidence of reads breaking cleanly at splice junctions?
- Is there a strand bias in the direction of mapping relative to the transcripts? If there is, then is it same or opposing strand specific? Is it consistent across all transcripts?
- Are there reads in introns / intergenic regions? If so are the amounts consistent across samples?
- Is there evidence of technical duplication (PCR artefacts)?

Preliminary QC

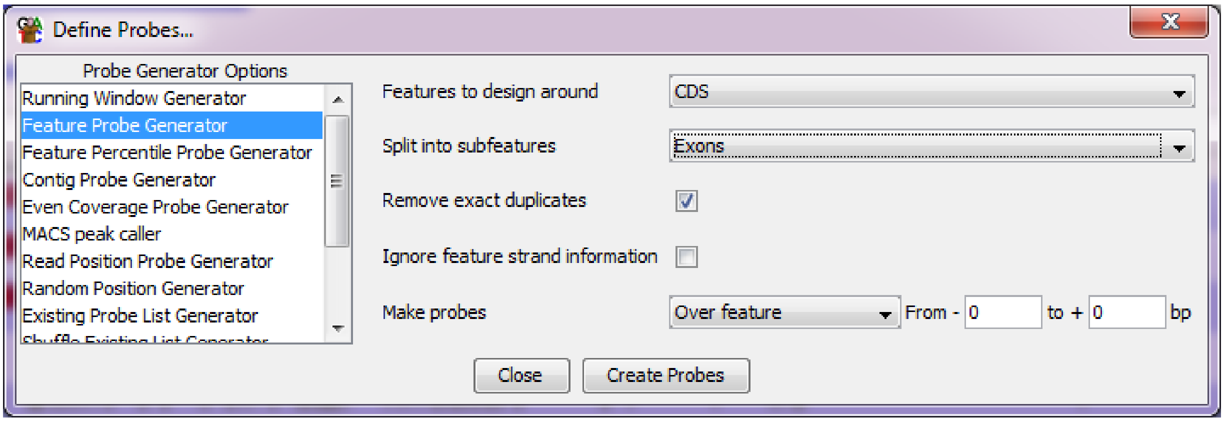
Before we get into the proper quantitation of our data we can look at some automated QC.

The first plot you can draw is an RNA-Seq QC plot. This will look at the proportion of reads falling into genes, the proportion of reads in genes falling into exons and the strand specificity of the library. It is important that these parameters are similar for all samples in your study. If they are not then you should try to understand whether the difference is due to a difference in a specific locus or a general change in your entire data. You can draw the plot using Plots > RNA-Seq QC plot and accepting the defaults.

Draw the plot and see if there are any samples which seem to be mis-behaving.

The other QC we can do is to look at the duplication levels in the samples. Because these are confounded by the high coverage in certain regions we can use a plot of duplication vs read density over all exons to see if we might have technical problems with the samples.

To make probes over all exons go to Data > Define Probes. You'll use the Feature Probe Generator to put probes over mRNA features, split into Exons.


Define Probes...
Probe Generator Options
Running Window Generator
Feature Probe Generator
Feature Percentile Probe Generator
Contig Probe Generator
Even Coverage Probe Generator
MACS peak caller
Read Position Probe Generator
Random Position Generator
Existing Probe List Generator
Shuffle Existing List Generator
Close
Create Probes

Features to design around: CDS

Split into subfeatures: Exons

Remove exact duplicates ☑

Ignore feature strand information □

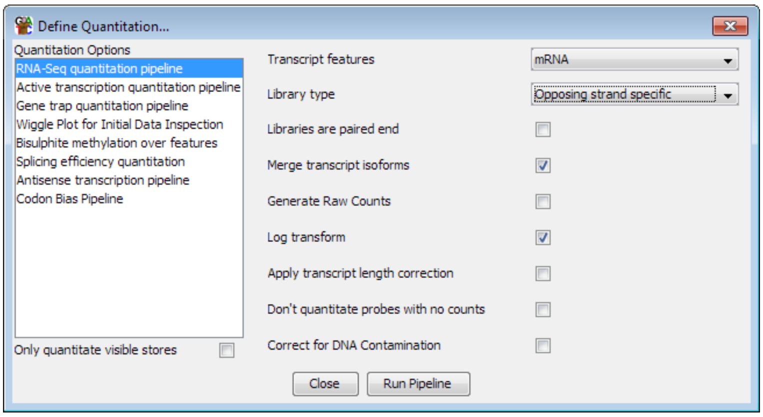
Make probes: Over feature From - 0 to + 0 bp

After making the probes you will be prompted to quantitate them. You can just use a read count quantitation on default settings.

To draw the duplication plot select Plots > Duplication Plot. You should see that there is a relationship between read density and duplication, with low density exons having relatively low duplication. If the plot shows everything with high duplication, or the distribution is not continuous then you may have a problem.

Performing a standard RNA-Seq quantitation

You can perform a standard quantitation of your RNA-Seq data by selecting Data > Quantitation Pipelines > RNA-Seq quantitation pipeline and then keeping the default options. The only thing you will need to change is the type of library. In this case the data is an opposing strand specific library (mapped reads are on the opposite strand to the feature they come from).


Define Quantitation...
Quantitation Options
RNA-Seq quantitation pipeline
Active transcription quantitation pipeline
Gene trap quantitation pipeline
Wiggle Plot for Initial Data Inspection
Bisulphite methylation over features
Splicing efficiency quantitation
Antisense transcription pipeline
Codon Bias Pipeline
Close
Run Pipeline

Only quantitate visible stores □

Transcript features: mRNA

Library type: Opposing strand specific

Libraries are paired end □

Merge transcript isoforms ☑

Generate Raw Counts □

Log transform ☑

Apply transcript length correction □

Don't quantitate probes with no counts □

Correct for DNA Contamination □

This will quantitate at the gene level by counting the number of reads which fall into exons of each gene and correcting for the total number of reads in the sample. The final quantitated values in this case would be reads per million reads of input, and will be log2 transformed.

Quantitation QC

After quantitating, the next step is to draw a distribution plot to check whether the distributions for your different samples are the same or not. Generally in RNA-Seq you expect that the overall distributions of counts should be similar, but that different individual genes may change expression. If the distributions are not well matched it either indicates a problem with your data, or that the default normalisation to total read count is not good enough for this data. You can draw a distribution plot by selecting Plots > Cumulative distribution plot > Visible Data Stores

If a subset of the lines is separated from the rest, and run parallel over most of their length then you can bring them back together using the percentile normalisation quantitation.
Data > Quantitate Existing Probes > Percentile Normalisation > Quantitate

If the lines for the different samples do not match up because some are spread at the low end of the plot then this could indicate the presence of DNA contamination. You could try to resolve this by re-running the quantitation above with the “Correct for DNA contamination” option turned on.

If you do any normalisation then draw the Cumulative Distribution plot again to check that the normalisation has improved.

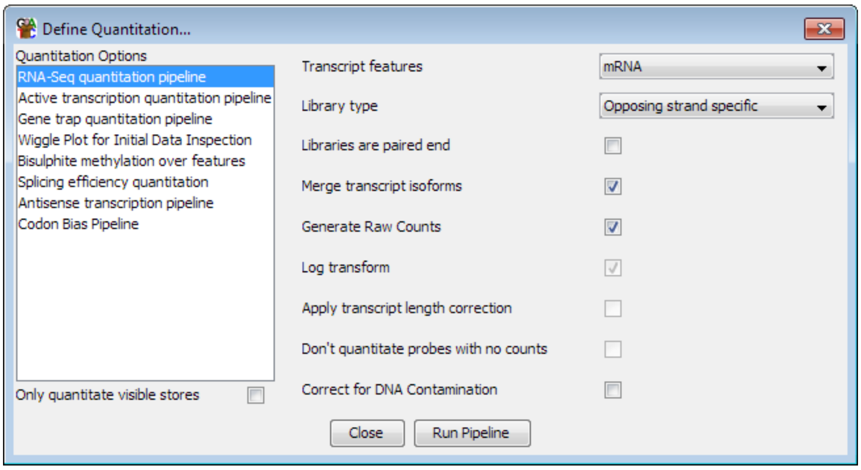
The other basic QC metric you can do is to look at the clustering of your samples. You will hopefully find that samples from the same biological condition cluster together. You may also see some outliers which might indicate a problem with some of your replicates and you may then choose to remove these before proceeding rather than including poor quality data in your analysis. To draw a cluster tree you can do:
Plots > Data Store Similarity > DataStore Tree

Then see if the grouping of the samples agrees with what you expect given their names. Are the groups equally related, or is one more tightly correlated than the other?

Generation of raw counts for DESeq2 analysis

DESeq2 requires a set of raw counts which have not been transformed or corrected. We’re therefore going to generate this list and run the analysis before doing a more visualisation friendly quantitation to review the results.

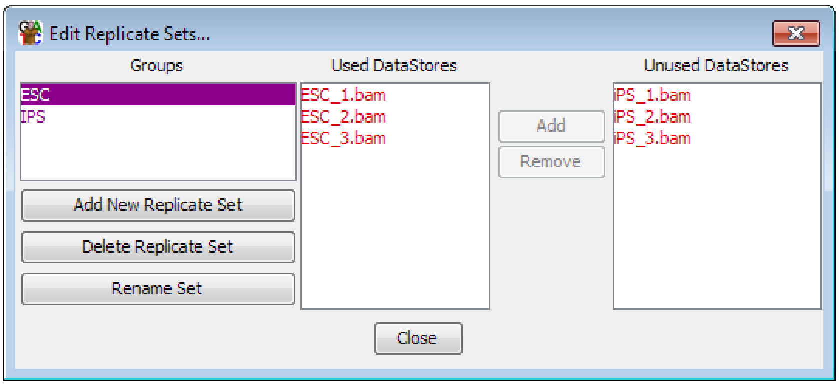
To generate raw counts go back to Data > Quantitation Pipelines > RNA-Seq quantitation pipeline and opt to quantitate using the ‘Generate Raw Counts’ option.


Define Quantitation...
Quantitation Options
RNA-Seq quantitation pipeline
Active transcription quantitation pipeline
Gene trap quantitation pipeline
Wiggle Plot for Initial Data Inspection
Bisulphite methylation over features
Splicing efficiency quantitation
Antisense transcription pipeline
Codon Bias Pipeline
Close
Run Pipeline

Transcript features: mRNA
Library type: Opposing strand specific
Libraries are paired end ☐
Merge transcript isoforms ☑
Generate Raw Counts ☑
Log transform ☑
Apply transcript length correction ☐
Don't quantitate probes with no counts ☐
Correct for DNA Contamination ☐
Only quantitate visible stores ☐

Generating replicate set groups for DESeq2 analysis

For DESeq2 analysis we need to tell the program which samples are part of the same biological condition group by creating Replicate Sets within SeqMonk. You do this by selecting Data > Edit Replicate Sets. You should create one set for the ESC samples and one for the IPS samples and then add the datasets to the appropriate replicate sets.



Once this is done you can change the view to show only the replicate sets by selecting View > Set Data Tracks and removing the datasets (in red) and adding the replicate sets (in purple).

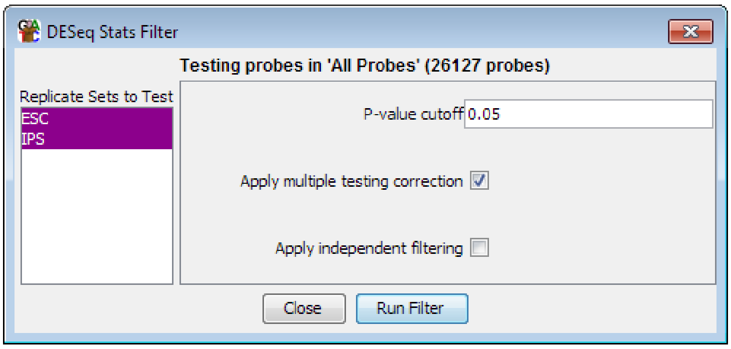
Exercise 5: Differential Expression analysis with DESeq2

Although DESeq2 is an R program, because SeqMonk can talk directly to R, we can run the DESeq2 analysis from within the SeqMonk interface. In this section we will run the DESeq2 analysis in SeqMonk, but will also include the R script so you can see what it is doing if you want to run it manually in future.

If we wanted to run this analysis in R we would create a report using Reports > Annotated Probe Report, and then save that to a file.

Running DESeq2 analysis in SeqMonk

To run your DESeq2 analysis select Filtering > Filter by Statistical Test > Count based statistics > Replicated Data > DESeq2. You should see the replicate sets you made in the last exercise on the left of the options and you should select both of these so we can compare them.



DESeq Stats Filter

Testing probes in 'All Probes' (26127 probes)

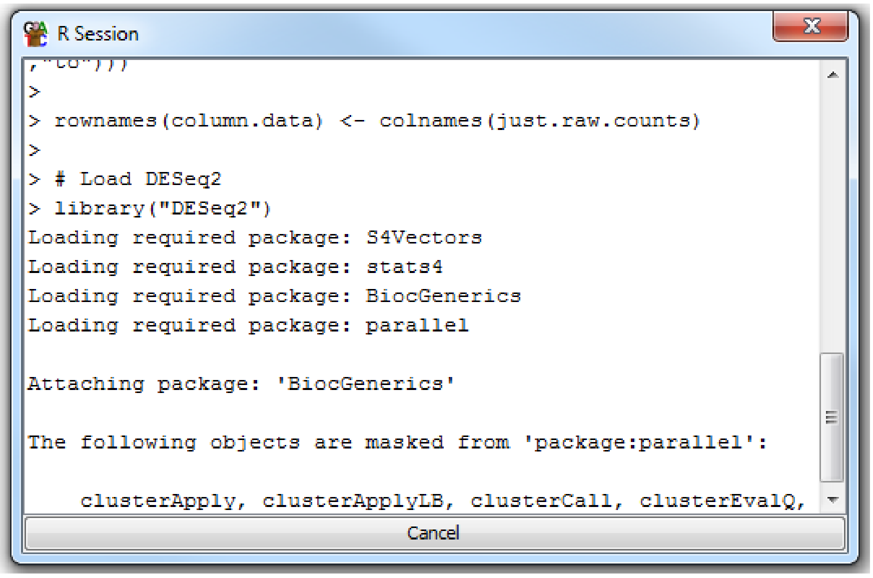
Replicate Sets to Test
ESC
IPS

P-value cutoff 0.05

Apply multiple testing correction ☑

Apply independent filtering ☐

When you run the filter you will see an R window open up and the script will scroll past. At the end you should see the hits returned as a normal SeqMonk hit list.



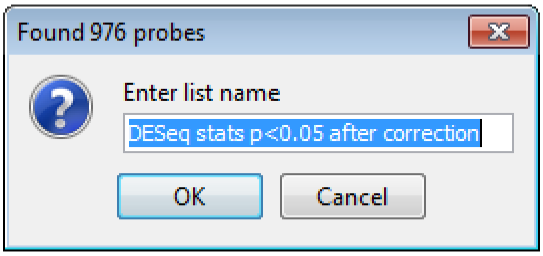
R Session

```
,"c0")))
>
> rownames(column.data) <- colnames(just.raw.counts)
>
> # Load DESeq2
> library("DESeq2")
Loading required package: S4Vectors
Loading required package: stats4
Loading required package: BiocGenerics
Loading required package: parallel

Attaching package: 'BiocGenerics'

The following objects are masked from 'package:parallel':

    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
```



Found 976 probes

Enter list name

DESeq stats p<0.05 after correction

Examine a standard DESeq2 analysis script

This section is for INFORMATION ONLY you DON'T NEED TO RUN ANYTHING here.

Below is an example of a standard DESeq2 analysis. If you're not likely to run DESeq2 manually you can skip over this section, but it might be useful to see what's going on behind the scenes so you can understand the steps taken. You don't need to actually run this code during the exercise.

First we set the directory where the count table can be found.

```
setwd("~/RNA-Seq_Course_Data/Results_Files/Mouse_ES_Data/")
```

Now we read in the count table into an R data frame.

```
raw.counts <- read.delim("per_gene_counts.txt",quote="")
```

We're going to use the gene names to identify our hits. The next line simply removes any instances where the same gene name appears more than once in the count table. There are still some genome annotations where multiple genes are given the same name, though this is slowly being sorted out.

```
raw.counts[!(duplicated(raw.counts$Probe) |
duplicated(raw.counts$Probe,fromLast=TRUE)),] -> raw.counts
```

We set the gene names to be the row names (index) for the data frame

```
rownames(raw.counts) <- raw.counts$Probe
```

Our original dataset has some extra columns of information in it (descriptions, other ids etc), which we don't need. This line selects just the columns with the counts in.

```
raw.counts[,13:18] -> just.raw.counts
```

We now need to make up a data structure which tells DESeq2 which experimental group each of our datasets is in. We build up a small table with the factors (ESC or IPS in this case) in one column, and the file names from the count table in the other.

```
column.data <-
data.frame(cell.type=as.factor(c("ESC","ESC","ESC","IPS","IPS","IPS")))
rownames(column.data) <- colnames(just.raw.counts)
```

Now we can load the DESeq2 library.

```
library("DESeq2")
```

We can now build DESeq2s own internal data structure from the count table and the condition table.

```
count.data.set <- DESeqDataSetFromMatrix(countData=just.raw.counts,
colData=column.data, design= ~ cell.type)
```

Now the data is prepared we can run the analysis. This does the normalisation, dispersion shrinkage and model generation.

```
count.data.set <- DESeq(count.data.set)
```

We can now retrieve the results from the analysis. We can specifty additional filters here which will affect the corrected p-value we determine.

```
binomial.result <-
results(count.data.set,cooksCutoff=FALSE,independentFiltering=FALSE)
```

Some genes won't have been analysed if they were filtered by the previous step, so the next line simply removes genes which weren't tested.

```
na.omit(binomial.result) -> binomial.result
```

From the full set of results we're now going to select only those whose corrected p-value was below 0.05.

```
binomial.result[binomial.result$padj <= 0.05,] -> significant.results
```

We can sort the results by p-value to put the most significant at the top.

```
significant.results[order(significant.results$padj),] -> significant.results
```

Finally we can write the table of results to a file

```
write.table(significant.results, sep=”\t”, file="deseq2_hit_genes.txt",
row.names=TRUE, col.names=NA, quote=FALSE)
```

Exercise 6: Reviewing DESeq2 results in SeqMonk

In this exercise we will check that the hits coming out of DESeq2 make sense, and can use the intensity difference filter to pick out the top hits for further study.

IMPORTANT Quantitating with normalised log2 RPM counts

Now that the count based statistics have been run we no longer want to use raw counts as our quantitation. We're going to be visualising the data so we MUST use log2 transformed normalised counts.

Can you therefore go back to Data > Quantitation Pipelines > RNA-Seq quantitation pipeline and rerun the quantitation turning OFF the "raw counts" option, so you again generate log2 RPM quantitation.

Visualising the hits in a scatterplot

One of the easiest ways of visualising a set of results is to highlight them on a scatterplot of the two conditions comparing the normalised log2 RPM values.

Before creating your scatterplot, first check that you have "All Probes" selected in the top left panel.
To create the scatterplot select:

Plots > Scatter Plot

To view the correct comparison you should then select your two replicate groups (the names in purple from the drop down menus. You should now be able to see a comparison of your two conditions.
To highlight the DESeq2 hits press the "Highlight Sublists" button then select the DESeq2 hits in the available lists list and press Add to move it to the selected lists side, then press OK.

You should now be able to see the full comparison with the DESeq2 hits highlighted. Check that the genes which have been selected look sensible.

Visualising the hits in a variation plot

As well as looking at the differences you can also look at the hits relative to the amount of variation they show within their replicate sets. Ideally you want your hits to show large differences between your replicate sets, but to have unremarkable levels of variation within each replicate set. Any hits which are outliers within a variation plot might indicate that the hits are coming largely from the DESeq2 dispersion shrinkage model, and might not be the ones you'd want to pick to follow up on.

You can create a variation plot by selecting:

Plots > Variation Plot

You will want to look at the STDEV values, and highlight the DESeq hits as before. Look in both the ES and iPS groups.

Visualising hit reproducibility with a heatmap

Another way to look at the hits you have generated is to use a heatmap to show not only the difference between the two sets of conditions, but also to show the consistency between the individual replicates.

To generate this plot you want to see the variability between the samples within the replicate sets. To do this select View > Data Track Display and then set Replicate Set Display to be Expanded.

To construct the heatmap **firstly select the DESeq2 hit probe list** in the top left panel, then select: Plots > Hierarchical Clusters > Per Probe Normalised

You should see a heatmap open up showing all of the replicates in columns, and the genes clustered together in rows. You can change the colour scheme using the drop down menu at the top (red green is fairly customary), and use the slider on the right to control the colour scaling.

Exercise 7: Combining DESeq and Intensity difference filtered lists

SeqMonk provides a different way to perform differential expression analysis which starts from a different premise to DESeq. You should have found that the DESeq results produced a list of several hundred hits, and from more highly powered experiments this list could easily comprise several thousand hits, this is because disruptions to biological systems will tend to disturb large numbers of genes as the whole system becomes destabilised, however there will hopefully be a set of genes which are changed more strongly as they directly respond to the stimulus.

The intensity difference filter in SeqMonk provides a way to identify changing genes by looking at the distribution of differences to find genes whose change isn't explained by the general level of disruption in the system. As such is can be used independently, or in conjunction with DESeq to provide a much smaller list of hits which might be easier to follow up.

Generating a list of intensity difference filtered genes

First, ensure that you have the "All Probes" list selected in the top left panel of SeqMonk.
In the SeqMonk project select Filtering > Filter by Statistical test > Continuous value statistics > Unreplicated data > Intensity Difference

Select the ESC replicate set (in purple) in the "From Data Store" list, and the IPS replicate set in the "To Data Store" list and press "Run Filter". Say OK to the name of the generated list.

Visualising the intensity difference hits

Construct a scatterplot of the two replicate sets (as you did for the DESeq hits) but highlight the intensity difference results. You can also try highlighting both the DESeq and intensity difference results (make sure DESeq is on top of intensity difference as it contains more genes), to get an idea of the differences between the two lists.

Create a combined hit list

Select Filtering > Logically Combine Existing Lists and create a new list which contains only genes found in both the DESeq and Intensity difference hit lists.

Draw a final scatterplot and variation plot which shows all 3 of these lists highlighted. Take a look at the data behind a few of the DESeq2 or SeqMonk specific hits to see if you can see why these would not have been found by both methods (you can double click on any point in the scatterplot to move the main view to that gene).

Export your final candidate list

Create an annotated probe report from your list of candidates identified by both DESeq2 and SeqMonk and annotate this with name matched genes.

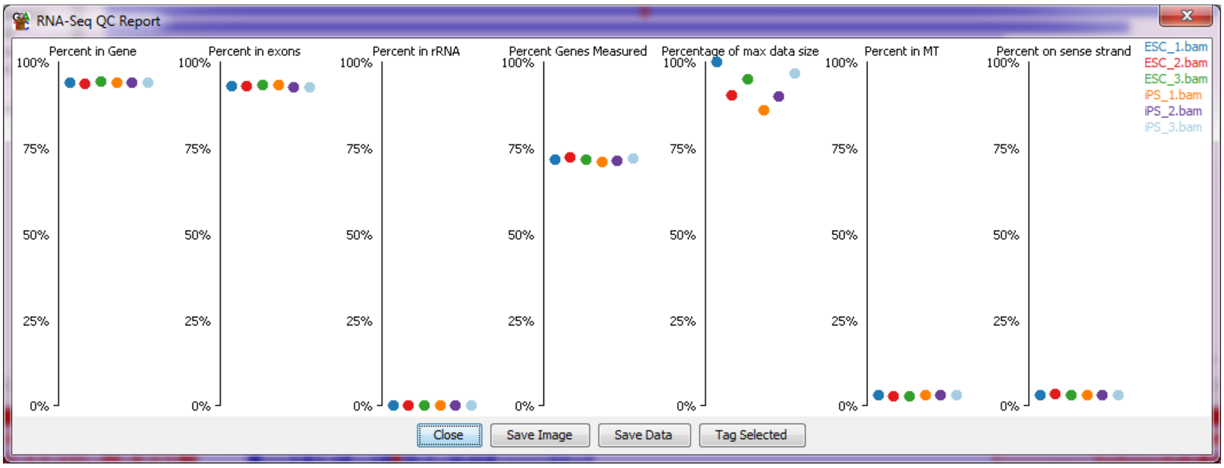
Example Plots

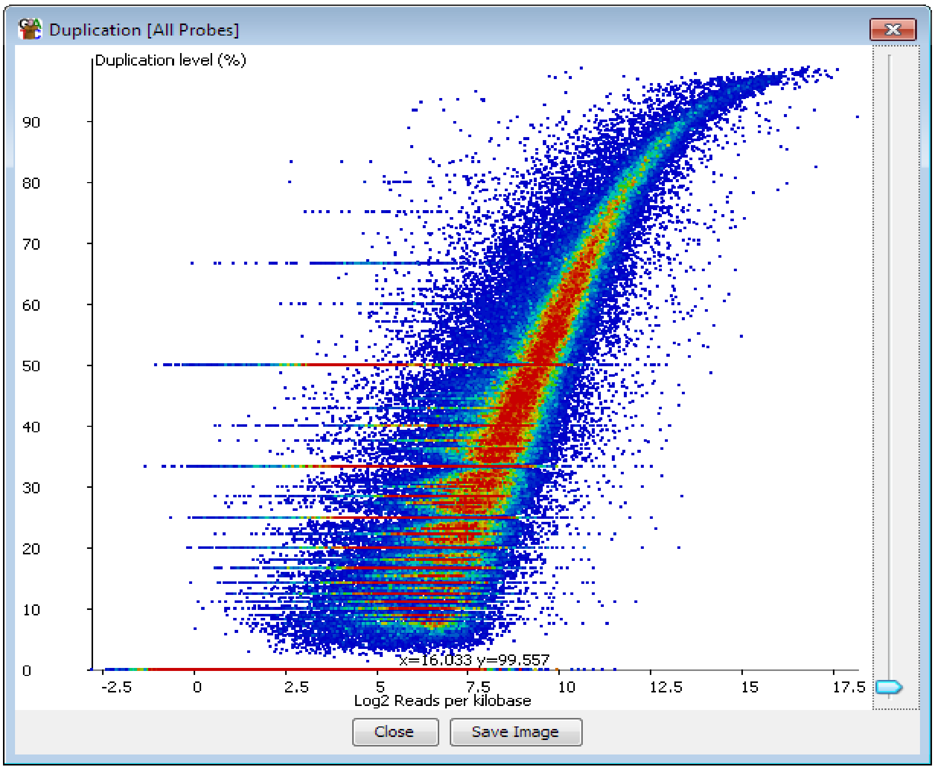
So you know what you should be seeing here are copies of the plots you should generate in this practical:

Hisat2 Mapping statistics

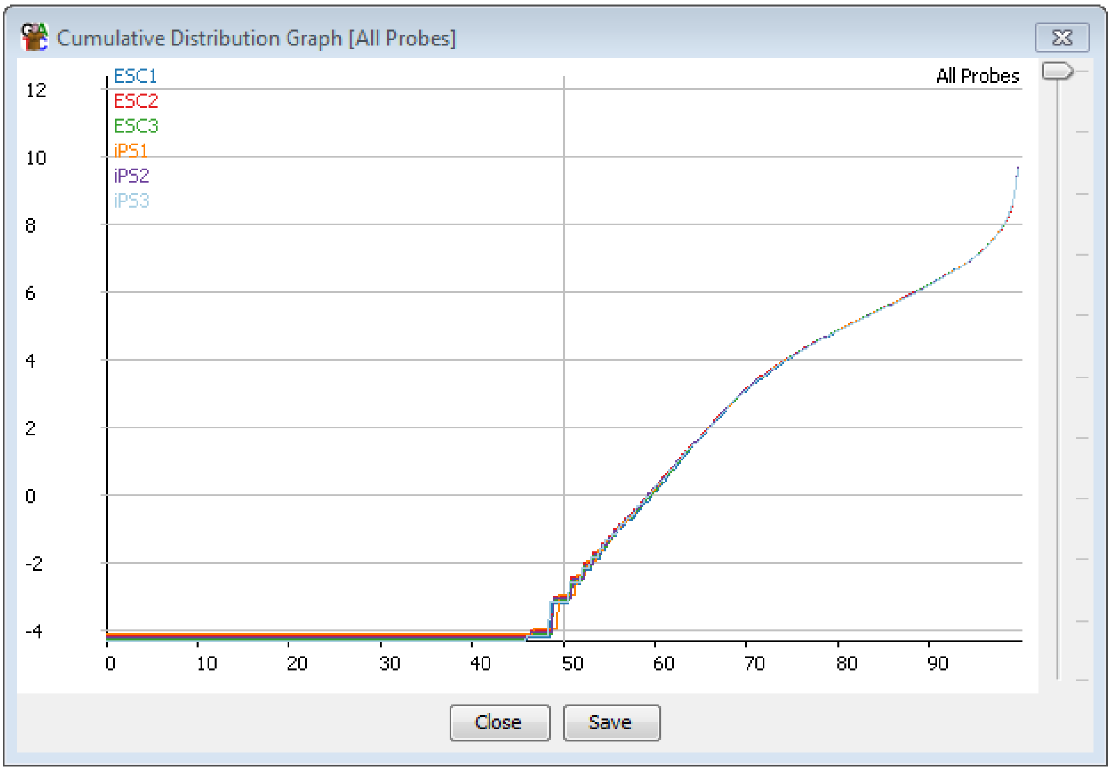
```
5725730 reads; of these:
  5725730 (100.00%) were unpaired; of these:
    616637 (10.77%) aligned 0 times
    4397277 (76.80%) aligned exactly 1 time
    711816 (12.43%) aligned >1 times
89.23% overall alignment rate
```

RNA-Seq QC Plot and Duplication Plot

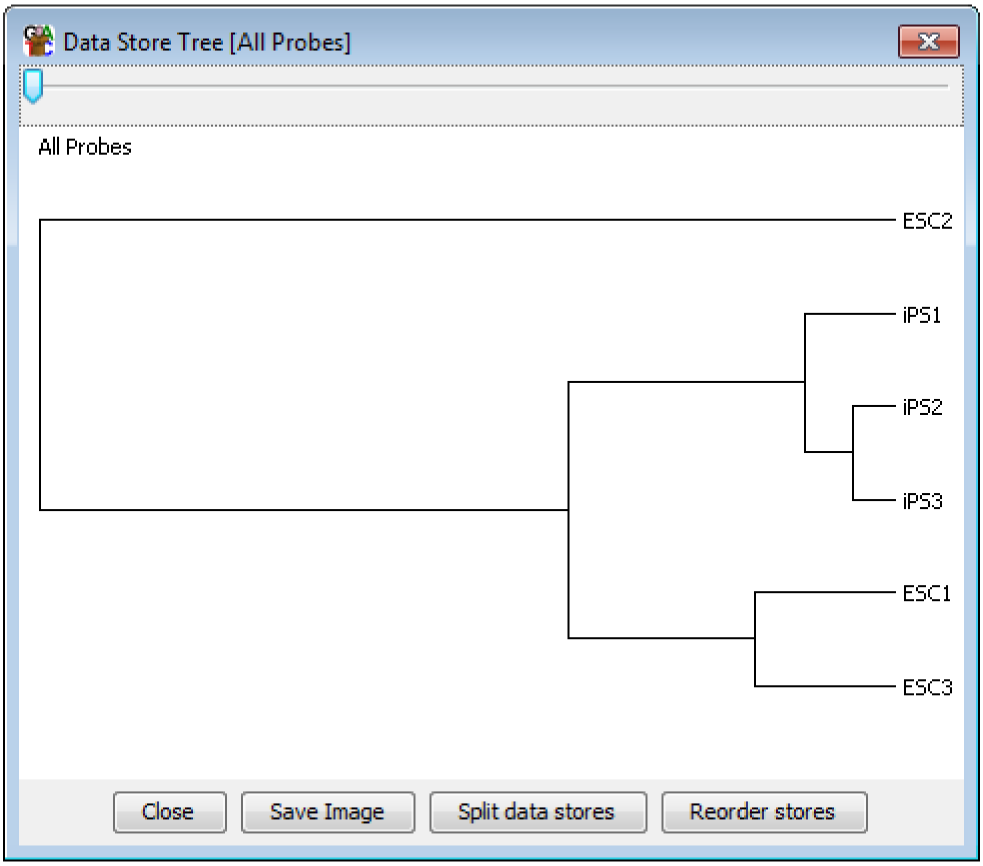




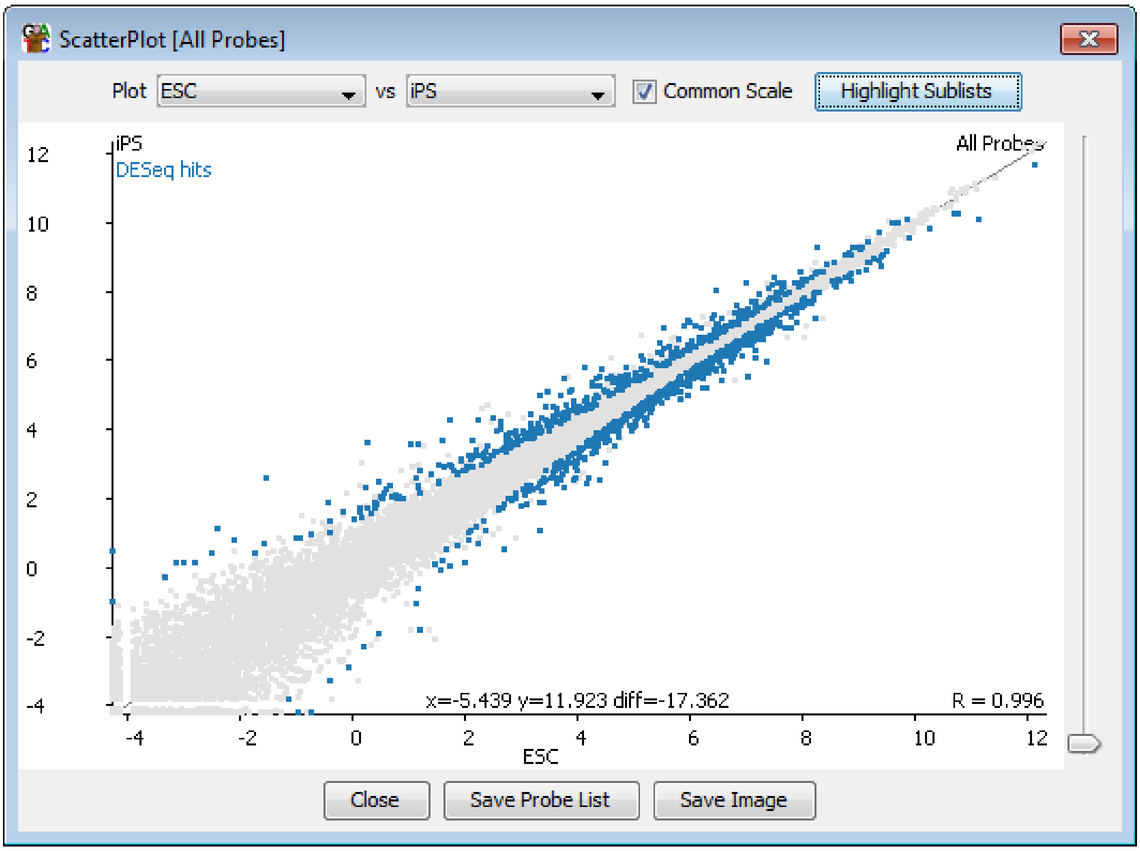
Cumulative Counts without normalisation (no normalisation is needed here)



Clustered Tree of Samples



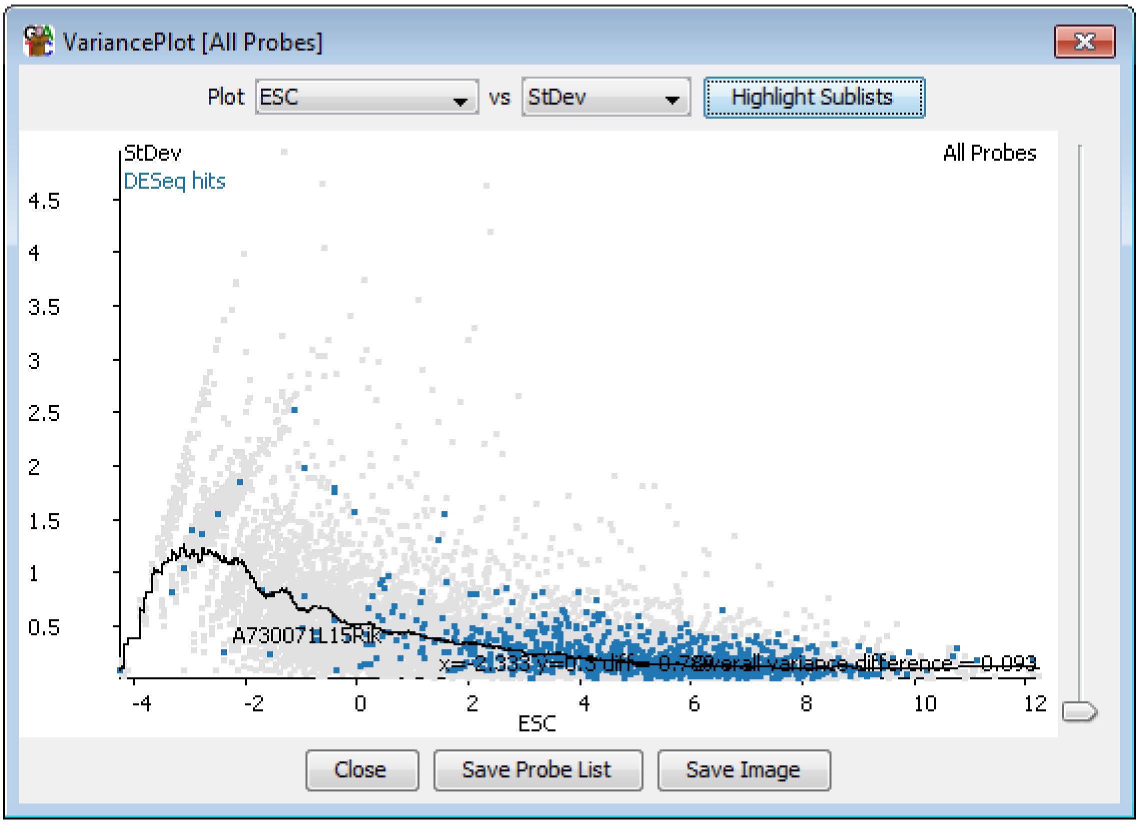
Visualisation of DESeq results in a scatterplot



ScatterPlot [All Probes]

Plot ESC vs iPS ☑ Common Scale

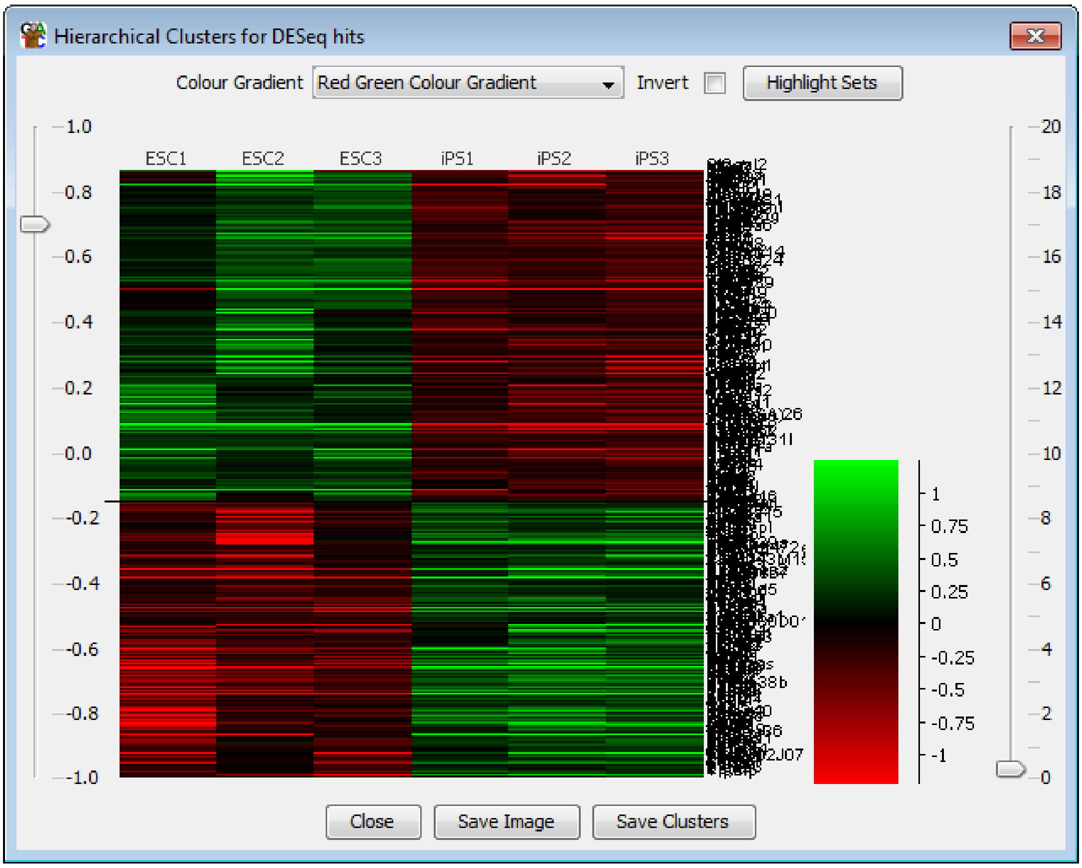
Visualisation of DESeq results in a variation plot



VariancePlot [All Probes]

Plot ESC vs StDev

Heatmap of replicates for DESeq results

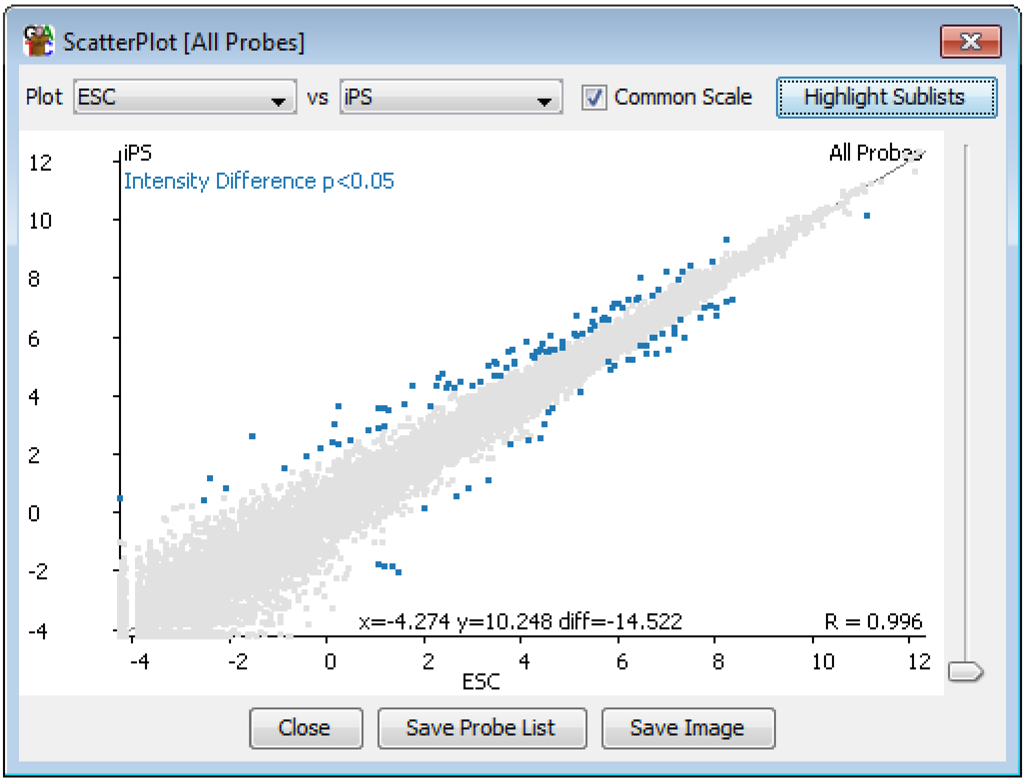


Hierarchical Clusters for DESeq hits

Colour Gradient: Red Green Colour Gradient

Invert □

Scatterplot highlighting intensity difference results



ScatterPlot [All Probes]

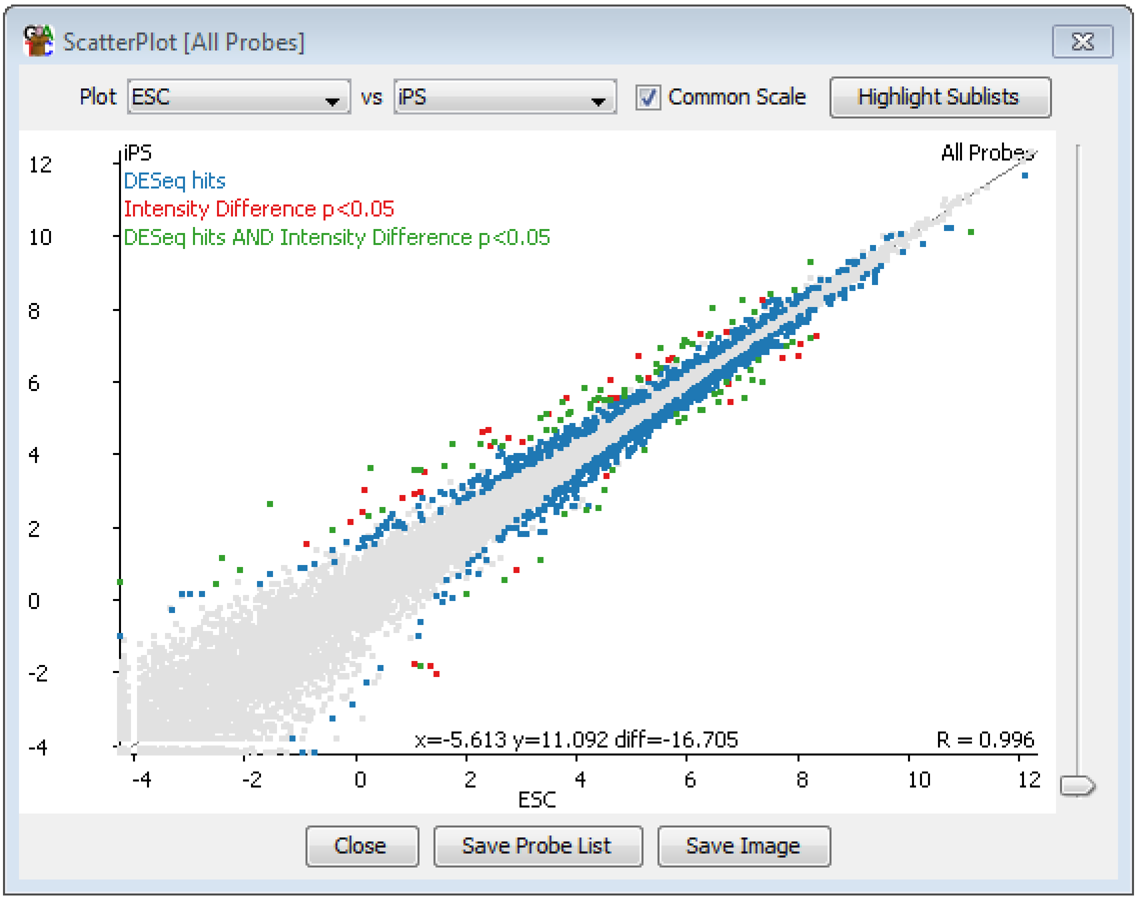
Plot ESC vs iPS

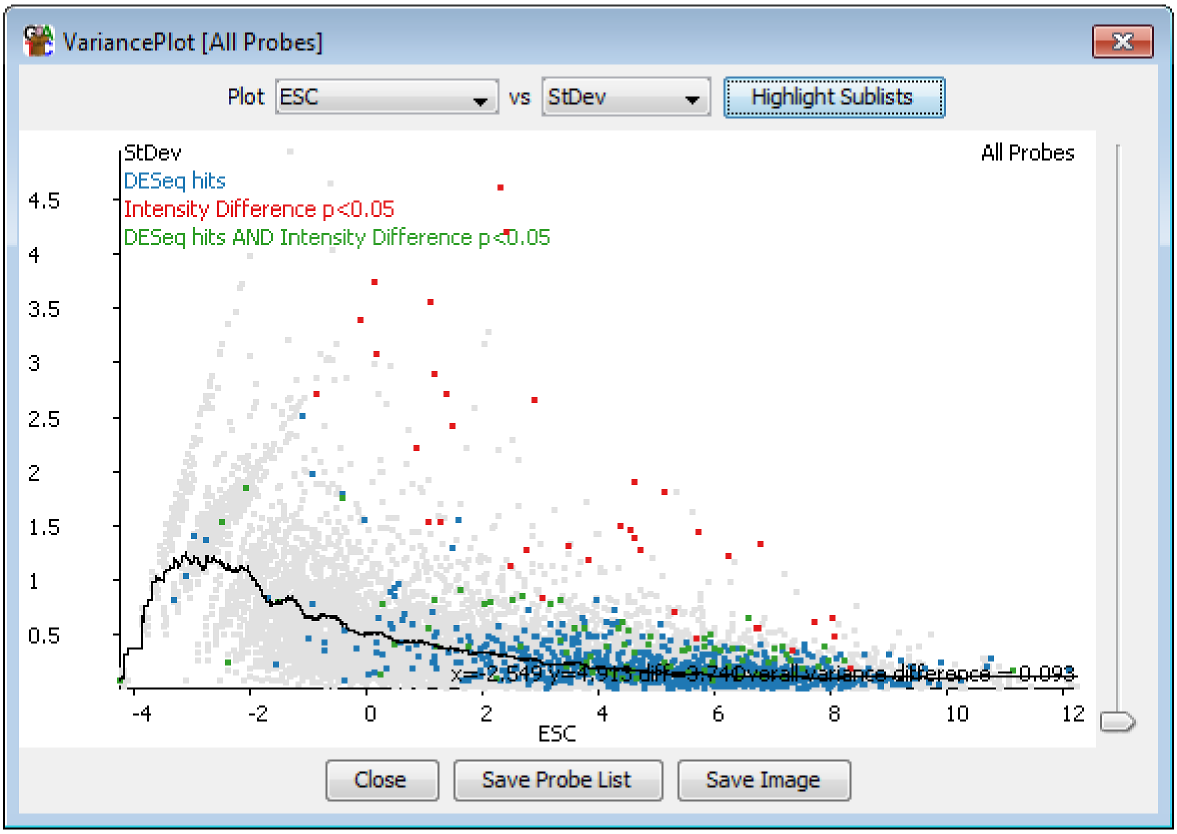
☑ Common Scale

x=-4.274 y=10.248 diff=-14.522

R = 0.996

Scatterplot and variation plot highlighting both DESeq and Intensity difference hits


ScatterPlot [All Probes]
Plot ESC vs iPS ☑ Common Scale Highlight Sublists
iPS
DESeq hits
Intensity Difference p<0.05
DESeq hits AND Intensity Difference p<0.05
All Probes
12
10
8
6
4
2
0
-2
-4
x=-5.613 y=11.092 diff=-16.705
R = 0.996
-4
-2
0
2
4
6
8
10
12
ESC
Close
Save Probe List
Save Image


VariancePlot [All Probes]
Plot ESC vs StDev Highlight Sublists
StDev
DESeq hits
Intensity Difference p<0.05
DESeq hits AND Intensity Difference p<0.05
All Probes
4.5
4
3.5
3
2.5
2
1.5
1
0.5
x=2.549 y=4.913 diff=3.740 overall variance difference = 0.093
-4
-2
0
2
4
6
8
10
12
ESC
Close
Save Probe List
Save Image